# *Applying Constraint Grammar to Tibetan NLP*

**Edward Garrett & Nathan Hill**

Project website: http://larkpie.net/tibetancorpus/

# Workflow – Some rules

### 020a. Disambiguating [n.rel] and [n.count]

Background: By definition [n.rel] cannot be quantified or modified, consequently, if a word has both [n.rel] and [n.count], and is quantified or modified, then the reading [n.rel] can be removed.

Rule: If a word 'w' has both [n.count] and [n.rel] as tags and this word is followed by a word with an unambiguous analysis [adj] or [num.ord] then delete the analysis [n.rel] from 'w'.

### 020b. Distinguishing [n.rel] from [n.count]

Background: By definition a relator noun is either preceded directly by a noun or by a genitive case marker. Thus, if a word that is either a relator noun or a count noun is preceded by an associative case marker, then the wordd in question must be interpreted as a count noun and not as a relator noun.

Rule: If a word has both the tags [n.count] and [n.rel] and is preceded by *dań*, then delete the tag [n.rel] from the word in question.

# Workflow – Flagging the rule

**Label:** 020b
**Title:** Distinguishing [n.rel] from [n.count]
**Background:**
By definition a relator noun is either preceded directly by a noun or by a genitive case marker. Thus, if a word that is either a relator noun or a count noun is preceded by an associative case marker, then the wordd in question must be interpreted as a count noun and not as a relator noun.
**Rule:**
If a word has both the tags [n.count] and [n.rel] and is preceded by *dań*, then delete the tag [n.rel] from the word in question.
**Pattern:**
((?:^|\s)ϥϵ\|\S+\s+\S+\|\S*\[n\.count\]\S*)\[n\.rel\](\S*)
**Replace:**
$1$2
**CG3:**
REMOVE (n.rel) (-1 ("<ϥϵ>")) (0 (n.count)) ;
**Rule tags:**
Version 2
Tell us if this rule needs attention  Tell us if this rule is broken

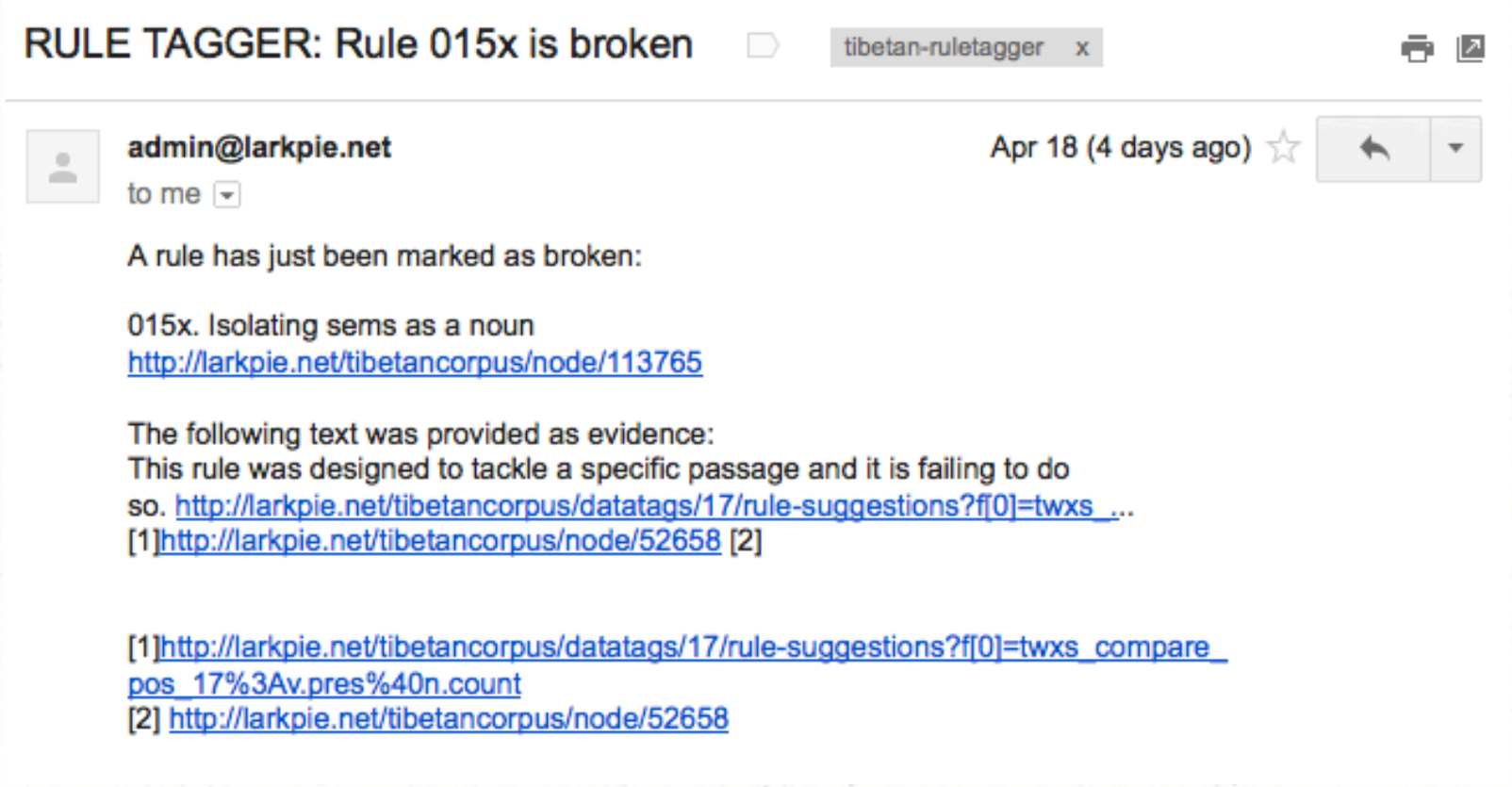
# Workflow – Flagging the rule

Please state how the rule has changed.

**Reason**

This rule needs to be rewritten to better handle that case that we were discussing over lunch.

# Workflow – Fixing the rule



RULE TAGGER: Rule 015x is broken

tibetan-ruletagger x

**admin@larkpie.net** Apr 18 (4 days ago)

to me

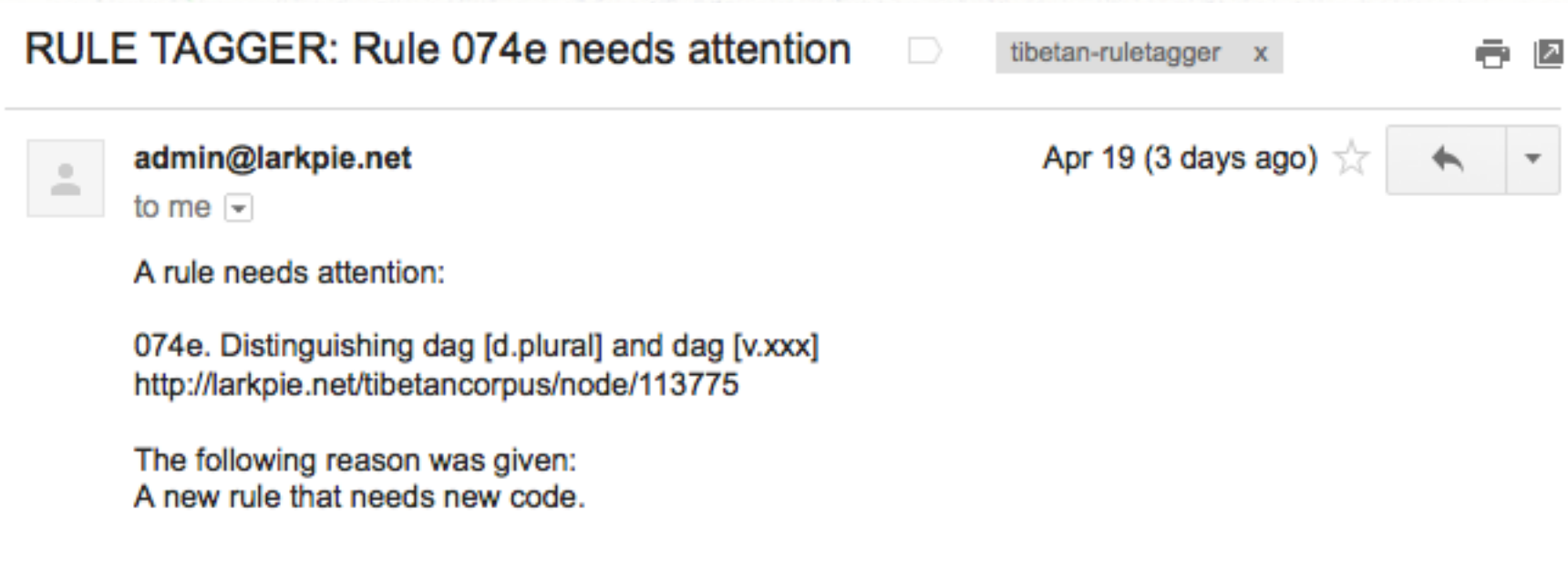
A rule has just been marked as broken:

015x. Isolating sems as a noun
http://larkpie.net/tibetancorpus/node/113765

The following text was provided as evidence:
This rule was designed to tackle a specific passage and it is failing to do so. http://larkpie.net/tibetancorpus/datatags/17/rule-suggestions?f[0]=twxs_...
[1]http://larkpie.net/tibetancorpus/node/52658 [2]

[1]http://larkpie.net/tibetancorpus/datatags/17/rule-suggestions?f[0]=twxs_compare_pos_17%3Av.pres%40n.count
[2] http://larkpie.net/tibetancorpus/node/52658

# Workflow – Fixing the rule


RULE TAGGER: Rule 074e needs attention

tibetan-ruletagger x

**admin@larkpie.net** Apr 19 (3 days ago)

to me

A rule needs attention:

074e. Distinguishing dag [d.plural] and dag [v.xxx]
http://larkpie.net/tibetancorpus/node/113775

The following reason was given:
A new rule that needs new code.

# Workflow – Fixing the rule

View | Edit | Revisions | Log | Devel

Search

Clone content

**Label:** 019a
**Title:** Precluding na as a verb
**Background:**
The syllable *na* either many things including the locative case marker, locative converb, and the verb 'be sick'. The locative converb will perforce appear after a verb, and often before a punctuation marker or a focus clitic. In the same context, the verb 'sick' would be unlikely. (It is hard to imagine how 'sick' could be preceded directly by a verb, if such an occasion does arise this rule can be modified). So, after a verb a the end of a clause, *na* can be precluded as a verb.
**Rule:**
I*f na* appears after a verb [v.xxx] and before punctuation [punc] or a focus clitic [cl.focus], then delete any [v.xxx] tags from *na.*
**Pattern:**
(\S+\|(?:\[v\.[^\]]*\])+\s+ན\|\S*?)(?:\[v\.[^\]]*\])+(\S*\s+\S+\|\[(?:cl\.focus|punc)\](?:$|\s))
**Replace:**
$1$2
**CG3:**
REMOVE v.xxx (-1C v.xxx) (0 ("<ན>")) (1 (cl.focus) OR (punc)) ;
**Rule tags:**
Version 2
Tell us if this rule needs attention  This rule is broken

# Workflow – Fixing the rule

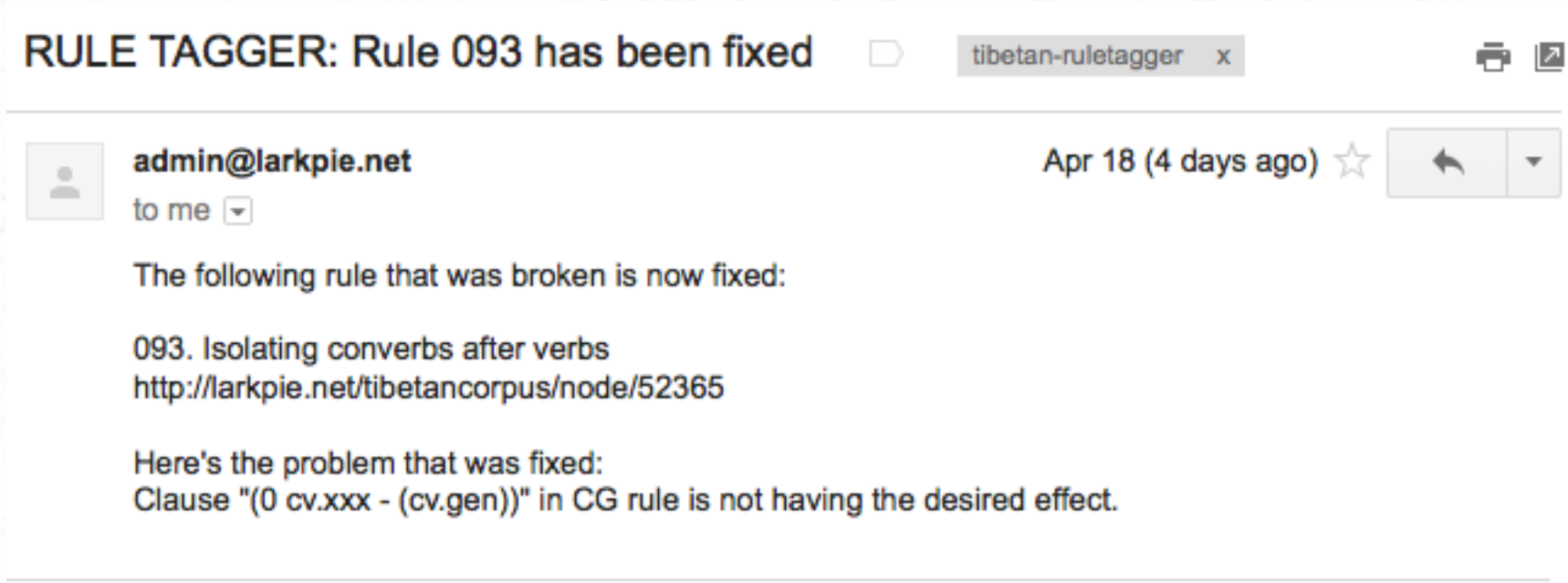
View | Edit | Revisions | Log | Devel

Search

Clone content

**Label:** 074e
**Title:** Distinguishing dag [d.plural] and dag [v.xxx]
**Background:**
The syllable *dag* has a number of possible tags, including [d.plural] and [v.xxx] 'be pure'. In the phrases such as *de dag gi dpe-cha* and *de dag daṅ de-bźin-gśegs-pa* it is clear that *dag* is not a verb.
**Rule:**
If *dag* follows *ḥdi* or *de* and precedes a *gi* (or *daṅ*) which is itself before an unambiguous [n.count] (i.e. *de dag gi* XXX|[n.count], *ḥdi dag gi* XXX|[n.count], *de dag daṅ* XXX|[n.count], and *ḥdi dag daṅ* XXX|[n.count]) then remove from *dag* any [v.xxx] tags.
**Rule tags:**
Version 2
This rule is being attended to  Tell us if this rule is broken

# Workflow – Fixing the rule



RULE TAGGER: Rule 093 has been fixed

tibetan-ruletagger x

**admin@larkpie.net** Apr 18 (4 days ago)

to me

The following rule that was broken is now fixed:

093. Isolating converbs after verbs
http://larkpie.net/tibetancorpus/node/52365

Here's the problem that was fixed:
Clause "(0 cv.xxx - (cv.gen))" in CG rule is not having the desired effect.

# Regex rules

The regex tagger consists of a sequence of rules, applied in order to a **horizontal text**. Each rule consists of two parts, the pattern (before the < symbol) and the replacement (after the < symbol).

In **horizontal format**, a single space marks the boundary between words, and line breaks separate sentences. Each word consists of a word form followed by a tag, with the pipe character in between. Whitespace is not permitted within words.

ར་|[case.term][cv.term][dunno][n.count][skt] བསྡུ་|[n.count][v.fut][v.fut.v.pres][v.imp][v.pres] ནས་|[case.ela] [cv.ela][dunno][n.mass] ཡབ་|[n.count][v.fut] ཀྱི་|[case.gen][cv.cont][cv.gen] ཞལ་ཆེམས་|[n.count] བཞིན་|[n.count] [n.rel] ཕ་ཤུལ་|[n.count] བུ|[n.count] ས་|[case.agn][cv.agn][dunno][n.count][n.mass][n.rel][skt] འཛིན་|[v.pres] དུ་| [case.term][cv.term] འཇུག་ས|[n.v.fut.n.v.pres][n.v.pres] ར་|[case.term][cv.term][dunno][n.count][skt] ཞུ་| [n.count][v.fut][v.fut.v.pres][v.imp][v.past][v.past.v.pres][v.pres] བྱས་ས|[n.v.past] ས|[case.agn][n.count][n.rel] །| [punc]

# Regex rules

**Label:** 020a
**Title:** Disambiguating [n.rel] and [n.count]
**Background:**
By definition [n.rel] cannot be quantified or modified, consequently, if a word has both [n.rel] and [n.count], and is quantified or modified, then the reading [n.rel] can be removed.
**Rule:**
If a word 'w' has both [n.count] and [n.rel] as tags and this word is followed by a word with an unambiguous analysis [adj] or [num.ord] then delete the analysis [n.rel] from 'w'.
**Pattern:**
(\S+\|\S*\[n\.count\]\S*)\[n\.rel\](\S*\s+\S+\|\[(?:adj|num\.ord)\](?:$|\s))
**Replace:**
$1$2

**Label:** 020b
**Title:** Distinguishing [n.rel] from [n.count]
**Background:**
By definition a relator noun is either preceded directly by a noun or by a genitive case marker. Thus, if a word that is either a relator noun or a count noun is preceded by an associative case marker, then the wordd in question must be interpreted as a count noun and not as a relator noun.
**Rule:**
If a word has both the tags [n.count] and [n.rel] and is preceded by *dań*, then delete the tag [n.rel] from the word in question.
**Pattern:**
((?:^|\s)ϥϽ̈\|\S+\s+\S+\|\S*\[n\.count\]\S*)\[n\.rel\](\S*)
**Replace:**
$1$2

# Regex rules

**Label:** 113a
**Title:** Prohibiting the imperative in non-finite and finite but explicitly non-imperative contexts
**Background:**
The imperative is generally not permitted before converbs, or other non-finite contexts (such as before *kyaṅ*). It is likely that further training data will prompt the inclusion of further contexts in which the imperative is impossible.
**Rule:**
If a word has more than one [v.xxx] tag, including [v.imp], and the following word either has the form *na*, *kyaṅ*, *yaṅ*, *nas*, *kyi*, or has any of the tags [cv.cont], [cv.ela], [cv.fin], [cv.impf], [cv.loc], [cv.ques], [cv.sem], or [cv.term], then remove the tag [v.imp] from the word in question.
**Pattern:**
(\S+\|\S*)(?:(\[v\.[^\]]*\])\[v\.imp\]|\[v\.imp\](\[v\.[^\]]*\]))(\S*\s+(?:(?:ན|ཀྱང|ཡང|ནས|ཀྱི)ˇ?\|\S+|\S+\|\S*\[cv\.(?:cont|ela|fin|impf|loc|ques|sem|term)\]\S*))
**Replace:**
$1$2$3$4

# Regex rules

**Label:** 116
**Title:** The prohibition of the past in the indirect infinite construction
**Background:**
As discussed in Garrett et al. (forthcoming c) matrix verbs appear never to subcategorize for the past stem in the indirect infinitive construction (e.g. *ltar ḥgro* 'go to see'). Consequently, the past stem can be precluded in this context. (Inevitably there are occasional exceptions such as *yeṅs ma gtam du byas so* / in the *Mdzaṅs blun*, because of cases of this type, it is necessary not to run the rule following *ma*.)
**Rule:**
If a word has more than one [v.xxx] tag, including [v.past], the preceding word is not *ma*, and the following two words are (1) a word with a possible [cv.term] tag, and (2) a word with a possible [v.xxx] or [n.v.xxx] tag, remove the tag [v.past] from the word in question.
**Pattern:**
((?:^|\s)(?!མ\|)\S+\s+\S+\|\S*)(?:(\[v\.[^\]]*\])\[v\.past\]|\[v\.past\](\[v\.[^\]]*\]))(\S*\s+\S+\|\S*\[cv\.term\]\S*\s+\S+\|\S*(?:\[(?:n\.)?v\.[^\]]*\])\S*)
**Replace:**
$1$2$3$4

# Regex rules – summary

Within short order, it became evidence that updating and maintaining regex rules would require a regular expressions wizard with a keen eye for slashes.

Those with the linguistic subject knowledge to write grammar rules for Tibetan are unlikely to also possess or wish to obtain the technical skills to write and maintain complex regular expressions.

The rule statements are immediately accessible to linguists, but the regex rules are not.

Moving forward, if we want to create a rule grammar framework that the Tibetan studies community can contribute to, perhaps regex rules are not the way to go.

# Constraint grammar – background

Constraint Grammar (CG) is a methodological paradigm for natural language processing (NLP). Linguist-written, context dependent rules are compiled into a grammar that assigns grammatical tags ("readings") to words or other tokens in running text. Typical tags address lemmatisation (lexeme or base form), inflexion, derivation, syntactic function, dependency, valency, case roles, semantic type etc. Each rule either adds, removes, selects or replaces a tag or a set of grammatical tags in a given sentence context. Context conditions can be linked to any tag or tag set of any word anywhere in the sentence, either locally (defined distances) or globally (undefined distances). Context conditions in the same rule may be linked, i.e. conditioned upon each other, negated, or blocked by interfering words or tags. Typical CGs consist of thousands of rules, that are applied set-wise in progressive steps, covering ever more advanced levels of analysis. Within each level, safe rules are used before heuristic rules, and no rule is allowed to remove the last reading of a given kind, thus providing a high degree of robustness.

Source: http://en.wikipedia.org/wiki/Constraint_Grammar

# Constraint grammar – background

The Constraint Grammar concept was launched by Fred Karlsson in 1990 (Karlsson 1990; Karlsson et al., eds, 1995), and CG taggers and parsers have since been written for a large variety of languages, routinely achieving accuracy F-scores for part of speech (word class) of over 99%.[1] A number of syntactic CG systems have reported F-scores of around 95% for syntactic function labels. CG systems can be used to create full syntactic trees in other formalisms by adding small, non-terminal based phrase structure grammars or dependency grammars, and a number of Treebank projects have used Constraint Grammar for automatic annotation. CG methodology has also been used in a number of language technology applications, such as spell checkers and machine translation systems.

Source: http://en.wikipedia.org/wiki/Constraint_Grammar

# Constraint grammar – cohorts and readings

```
"<རྒྱང་ཚོད་ཀར་ཆུན>"
  "རྒྱང་ཚོད་ཀར་ཆུན" n.prop
"<མ་སྣད>"
  "མ་སྣད" n.count
"<ཚོར་སློངས>"
  "ཚོར་སློངས" n.count
"<བྱེད་པ>"
  "བྱེད་པ" n.v.pres
"<ཡིན་པ>"
  "ཡིན་པ" n.v.cop
"<འདུག>"
  "འདུག" v.fut
  "འདུག" v.fut.v.pres
  "འདུག" v.imp
  "འདུག" v.invar
  "འདུག" v.past
  "འདུག" v.past.v.pres
  "འདུག" v.pres
"<ཟེར་བ>"
  "ཟེར་བ" n.v.fut
  "ཟེར་བ" n.v.fut.n.v.pres
  "ཟེར་བ" n.v.past
  "ཟེར་བ" n.v.pres
"<བྱུང>"
  "བྱུང" v.imp
  "བྱུང" v.past
"<|>"
  "|" punc
```

# Constraint grammar – rules

#020a: Disambiguating [n.rel] and [n.count]

REMOVE (n.rel) (0 (n.count)) (1C (adj) OR (num.ord)) ;

#020b: Distinguishing [n.rel] from [n.count]

REMOVE (n.rel) (-1 ("<55>")) (0 (n.count)) ;

# Constraint grammar – rules

#113a: Prohibiting the imperative in non-finite and finite but explicitly non-imperative contexts

REMOVE (v.imp) (0 v.xxx - (v.imp)) (1 ("<(ན|ཙང|ཡང|ནས|ཀྱི)>"r) OR (cv.cont) OR (cv.ela) OR (cv.fin) OR (cv.impf) OR (cv.loc) OR (cv.ques) OR (cv.sem) OR (cv.term)) ;

#116: The prohibition of the past in the indirect infinite construction

REMOVE (v.past) (NOT -1 ("<མ>")) (0 v.xxx LINK NOT 0 (v.past)) (1 (cv.term)) (2 verbal) ;

# Constraint grammar – rules

#007: Limiting verb stems to single syllable

REMOVE v.xxx (0 ("<.+˙.+>"r)) ;

#009: Removing the 'dunno' tag

REMOVE (dunno) ;

#013: Distinguishing ches [v.past] from ches [adv.intense]

REMOVE (adv.intense) (0 ("<ཆེས་>")) (NOT 1 (adj) OR verbal) ;

#072c: Isolating raṅ as [d.det]

SELECT (d.det) (-1C (adj)) (0 ("<རང་>")) (1 ("<ཞིག་>")) ;

# Constraint grammar – rules

**016xc. Isolating re as a number**

BACKGROUND: The syllable *re* has many possible analyses. In the phrase *źo re źo do* 'one or two ounces' we tag it [num.card]. We presume that this construction is generalizabel to a pattern X *re* X *do*.

RULE: In the pattern X *re* X *do*, tag *re* as [num.card].

**016xd. Isolating re as a number**

BACKGROUND: The syllable *re* has many possible analyses. In the phrase *źo re źo do* 'one or two ounces' we tag it [num.card]. We presume that this construction is generalizabel to a pattern X *re* X *do*.

RULE: If *re* occurs otherwise than in the pattern X *re* X *do*, remove the tag [num.card] from *re*.

#016xc: Isolating re as a number

SELECT KEEPORDER (num.card) (-1 ("<(.+)>"r)) (0 ("<རེ>")) (1 ("<$1>"v)) (2 ("<དོ>")) ;

#016xd: Isolating re as a number

REMOVE KEEPORDER (num.card) (-1 ("<(.+)>"r)) (0 ("<རེ>")) (NEGATE 1 ("<$1>"v) LINK 1 ("<དོ>")) ;

Note: KEEPORDER “prevents the re-ordering of contextual tests”.

# Constraint grammar – complexities

**Label:** 028b
**Title:** Specifying źu as not [v.past] after nominalized verbs in the terminative
**Background:**
The syllable *źu* can either be the present or future of the verb 'request' or the past of a verb 'melt'. One often requests that someone do something, and in Tibetan this can be expressed by e.g. *ḥgro-bar źu* 'ask to go'. Thus, after nominalized verbs in the terminative case (always *-r* after *-pa/-ba*) the syllable *źu* can be specified as the verb 'reply'.
**Rule:**
In the pattern XXX[n.v.xxx] ར་ ཞུ་(ཏ་) delete [(n.)v.past] from ཞུ་(ཏ་). (In which XXX ends with -ཡ or -ཏ without a final tsheg).
**Pattern:**
(\S+[ཡཏ]\|(?:\[n\.v\.[^\]]*\])+\s+ར་\|\S+\s+ཞུ(?:་ཏ)?་?\|\S*)\[(?:n\.)?v\.past\](\S*)
**Replace:**
$1$2
**CG3:**
REMOVE past (-2 n.v.xxx LINK 0 ("<.*[ཡཏ]>"r)) (-1 ("<ར་>")) (0 ("<ཞུ(་ཏ)?་?>"r)) ;

# Constraint grammar – complexities

#037e: Finding words that are homophonous with forms of the final converb

REMOVE (cv.fin) (0 (n.count)) (1C gen) ;

#039b: Isolating the semi-final converb before śad

REMOVE (d.dem) (-1C v.xxx) (0 (cv.sem)) (1 shad) ;

#046: Isolating relator nouns that look like verbs

REMOVE v.xxx (-1 (case.gen)) (0 (n.rel)) ;

```
SET gen = ("<(དེ|ཀྱི|གི|གྱི|ཡི)˙?>"r) ;
SET shad = ("<[།ཿ]+>"r) ;
```

```
LIST v.xxx = v.aux v.cop v.cop.neg v.fut v.imp v.neg v.past v.pres ;
```

# Constraint grammar – complexities

#075b: Isolating pronouns in clause initial position

REMOVE cv.xxx (-1 shad.or.g) (0 p.xxx) ;

```
SET shad = ("<[།᛫᛬᛭ᛮᛯᛰ]+>"r) ;
SET tshegful = ("<.+་>"r) ;
SET tshegless = ("<.*[^་]>"r) - shad;
SET tshegless.g = ("<.*ག>"r) ;
SET tshegless.g.exception = ("<ད?ག>"r) ;
SET shad.or.g = shad OR tshegless.g ;
SET verba.dicendi = ("<(ཞེས|སྨོས|ཟེར|ཞུ|གསུངས|གསུང|ཞུས)>"r) ;
```

# Constraint grammar – complexities

#128: The creation of the tags [v.invar] and [n.v.invar]

APPEND ("$1"v v.invar) TARGET ("<(.*)>"r) (0 (v.fut) LINK 0 (v.past) LINK 0 (v.pres)) ;

APPEND ("$1"v n.v.invar) TARGET ("<(.*)>"r) (0 (n.v.fut) LINK 0 (n.v.past) LINK 0 (n.v.pres)) ;

REMOVE fut OR past OR pres (0 invar) ;

Variable string tags are in the form of "string"v, "<string>"v, and <string>v, where variables matching $1 through $9 will be replaced with the corresponding group from the regular expression match. Multiple occurances of a single variable is allowed, so e.g. "$1$2$1"v would contain group 1 twice.

# Constraint grammar – reservations

No steering committee oversight of CG syntax.

Open-source, but only one CG-3 implementation so far (in C).

Platform-specific building of C code may prove problematic for individual users.

CG on the web?

# Constraint grammar – extensions

```
"<nga>"
   "nga" p.pers B-NP @agn #1->5
"<yis>"
   "yis" case.agn I-NP
"<mi>"
   "mi" n.count B-NP @abs #3->5
"<maṅ-po>"
   "maṅ-po" adj I-NP
"<bsad>"
   "gsod" v.past O #5->0
```

**IOB Tagging**. The first word of a noun phrase is tagged B-NP for "begin NP", and subsequent words (if any) are tagged I-NP for "inside NP". Words that are outside chunks are tagged O. Thus, a full NP chunk consists of a B-NP tag followed by zero or more I-NP tags.

**Dependency Tagging**. Words are numbered from 1-5, with 0 representing the abstract sentence root. The parent of the verb *gsad* is the sentence root (#5->0), and its children are *nga* (#1->5) and *mi* (#3->5). Additional tags show the case frame role of these words: *nga* is in agentive case (@agn), and mi in absolutive case (@abs). Dependency relations can be profitably modeled within a system that assigns and manipulates tags at the level of the word.